Forecasting residential gas consumption with machine learning algorithms on weather data

Brian de Keijzer[1], Pol de Visser[1], Víctor García Romillo[2], Víctor Gómez Muñoz[3], Daan Boesten[1], Megan Meezen[1], Tadeo Baldiri Salcedo Rahola[1]

[1]*Faculty of Technology, Innovation and Society, The Hague University of Applied Sciences, Rotterdamseweg 137, 2628 CN, Delft, The Netherlands*

[2]*Faculty of Engineering, University of the Basque Country, Paseo Rafael Moreno 3 48013 Bilbao, Vizcaya, Spain*

[3]*Escuela Politécnica Superior, University Francisco de Vitoria, Carretera Pozuelo a Majadahonda, Pozuelo de Alarcón, Madrid, Spain*

11-01-2019, The Hague, The Netherlands

Abstract

Machine learning models have proven to be reliable methods in the forecasting of energy usage in commercial and office buildings. However, little research has been done on energy forecasting in dwellings, mainly due to the difficulty of obtaining household level data while keeping the privacy of inhabitants in mind. Gaining insight into the energy consumption in the near future can be helpful in balancing the grid and insights in how to reduce the energy consumption can be received. In collaboration with OPSCHALER, a measurement campaign on the influence of housing characteristics on energy costs and comfort, several machine learning models have been compared on forecasting performance and the computational time needed. Nine months of data containing the mean gas consumption of 52 dwellings on a one hour resolution has been used for this research. The first 6 months have been used for training, whereas the last 3 months have been used to evaluate the models. The results show that the Deep Neural Network (DNN) performs best with a 50.1 % MAPE on a one hour resolution. When comparing daily and weekly resolutions, the Multivariate Linear Regression (MVLR) outperforms other models, with a 20.1 % and 17.0 % MAPE, respectively. The models have been programmed in Python and are found at GitHub: https://github.com/deKeijzer/KB-74-OPSCHALER

Keywords: Forecasting – dwellings – natural gas consumption – neural networks – artificial intelligence – deep learning – machine learning – MVLR – DNN – GRU – LSTM – CNN – RNN– smart meter

1. INTRODUCTION

In recent years the European Commission has set CO_2 emission reduction targets. As a consequence, the Dutch government has been increasingly regulating the energy sector. The Netherlands aims to be free of natural gas usage by the end of 2050 [1]. Therefore, the importance of diminishing natural gas consumption in dwelling is growing.

Gaining insight in the prediction of gas consumption in dwellings is critical to meet the Dutch government requirements. Consequently, a vast number of researches has been conducted towards energy consumption prediction in commercial and office buildings [2, 3, 4, 5, 6]. Thus far, little research has been conducted to predict gas consumption on single household level due to the difficulty of obtaining the energy use data due to data privacy [7]. From commercial and office building energy forecasting research, Alberto Hernandez Neto [3] concludes that deep neural networks outperform physical simulation models by 3-6 %. Furthermore, according to Jurado López [8], weather conditions have the highest correlation to gas usage. It is worth noting that using multiple weather parameters as features in prediction models, improves the accuracy of the model [3].

This paper focusses on comparing the accuracy of widely discussed machine learning algorithms on an hourly, daily and weekly resolution and ultimately, creating a model architecture that can predict the natural gas usage of dwellings as accurate as possible. Different prediction models are tested, to evaluate their results. This is done with a dataset containing nine months of the mean hourly gas consumption data from a total of 52 dwellings in The Netherlands, together with the mean hourly weather data of a nearby weather station. In this paper the models use as less features as possible, to minimize the computation power required, which enables the models to be able to run on computers with limited computer power in dwellings.

2. METHODOLOGY

This paper focusses on the evaluation of different machine learning models. Specifically, Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and Convolutional Neural Network (CNN). These deep learning algorithms show the best potential for this type of prediction [1, 5, 9, 10, 4]. In most of the previously listed references, the models are based on electricity prediction of commercial and office buildings. These algorithms are called complex models in literature. The accuracy of the models is dependent on the environment of the research and chosen features. Consequently, the models need to be verified on residential households because this environment is different from commercial buildings. Commercial buildings usually have a fixed energy usage time schedule, while dwellings have more variation in its energy usage pattern [8].

In addition to the complex models, Multivariate Linear Regression (MVLR) and Deep Neural Network (DNN) show promising results in comparison to their simplicity [5, 11]. These models are called simple models in literature. The different models will be discussed in their respective subchapters.

In this section the way the data has been collected and processed will be discussed. Insights are given on what machine learning methods have been applied and which metrics are used for the evaluation of the models. The same metrics are used to evaluate the different machine learning models. Furthermore, this section will clarify the commonly used environmental settings for all the models.

2.1 Data collecting and processing

The data used in this research has been gathered by the OPSCHALER project from 2017-02 to 2017-12. It contains data concerning gas and electricity consumption. This information has been obtained from the smart meters of 52 different dwellings, everything is anonymized and therefore the dwelling locations are unknown. The data acquisition period from the dwellings varies from one to nine months. The electricity data is sampled with a ten second resolution, whereas the gas consumption is sampled with an hour resolution. These sampling periods have been visualized in *Figure 1*.

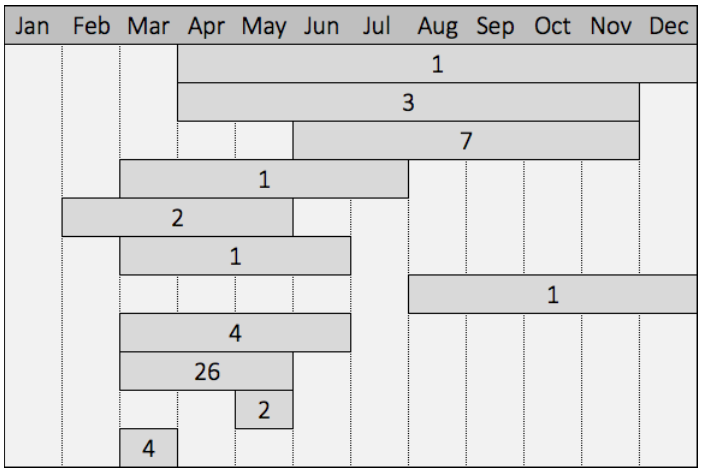

Figure 1. Distribution of the data acquisition from all different dwellings. The number in each grey selected area is the number of dwellings of which data has been gathered from during this period.

Due to an uneven and scarce distribution of the data, the mean of all dwellings at each timestamp is taken along with its standard deviation. This represents the mean gas usage on the aggregated level, e.g. a block of 52 dwellings.

In addition to this, weather information obtained from the Royal Netherlands Meteorological Institute (KNMI) station in Rotterdam has been used.. This is the weather station most nearby to all dwellings. However, the distance between this most distant dwelling is 103 km. The weather information has been sampled with a 15 minute resolution and contains different parameters. All available parameters can be found in *Appendix 1, Table 3*.

One of the goals of this research is to use as less features as possible, therefore a selection of the used parameters must be made[1]. According to [13, 14] air temperature and calendar related features are the most relevant to use when predicting gas consumption in the residential sector. In this research target is called 'gasPower', being this the gas consumption in an interval of an hour and measured in $[\mathrm{m}^3/\mathrm{h}]$. As a baseline, the parameter which has the highest Pearson correlation coefficient P with 'gasPower' is selected as a feature. In this case this parameter is the temperature T. The other parameters that have a Pearson correlation coefficient with the temperature that suffixes $|P| < 0.1$, are selected. Setting this threshold to $|P| < 0.1$ minimizes the influence of the features on each other. The Pearson correlation coefficient matrix of available parameters, including a descriptive table of them can be found in *Appendix 1, Figure 8 Table 3*. All the used parameters are visible in *Table 1.*

Table 1. Features extracted from the available parameters in *Appendix 1, Table 1*.
*Values have been calculated

Features	Unit	Description
FF	$\mathrm{m/s}$	Windspeed at 10 m
RG	$\mathrm{mm/h}$	Rain intensity
T	$^\circ\mathrm{C}$	Temperature at 1,5 m (1 minute mean)
hour of day	-	Hour of day at given the timestamp*.
day of week	-	Day of week at given the timestamp*.
season	-	Season of the year at given the timestamp*.

The weather data has been sampled with a 15 minutes resolution, whereas the 'gasPower' has been sampled with a one hour resolution. To combine these two datasets into one, the weather data has been down sampled to one hour by mean and combined with the 'gasPower'. Not a Number (NaN) values will appear in the intervals of time where the data acquisition system has stopped gathering information, this is the case for the features and target. All NaNs are removed from the dataset.

[1] A distinction between parameters and features is made. Parameters refer to all the available variables from the original dataset, whereas features refer to the variables that have been selected to forecast the target 'gasPower'.

2.2 Train, test & validation dataset

All models use the same train, validation and test dataset. A visualisation of the distribution of the train, validation and test dataset is shown in *Figure 2*. The first 70 % of the dataset is used as the train set. The test set containing the remaining 30 % is used for cross-validation of each model. During training of the neural networks, the last 20 % of the train set is taken as the validation set.

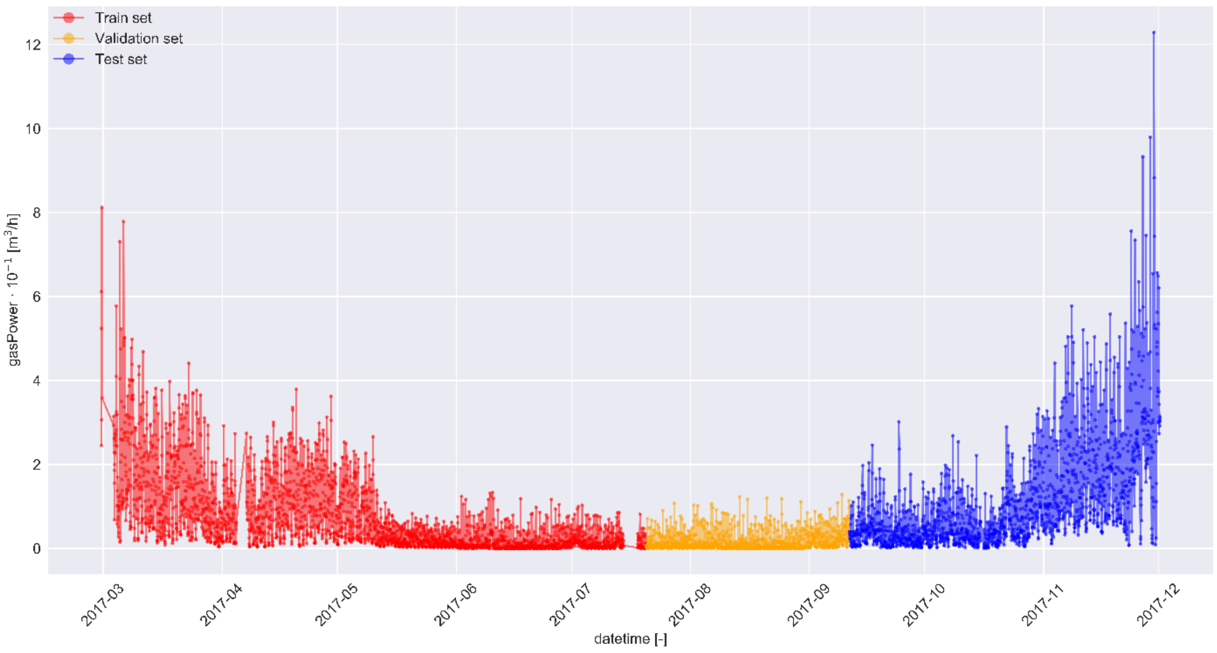

Figure 2. A visualization of the train, validation and test dataset distribution on a daily resolution.

2.3 Model evaluation

Neural networks use an optimizer to minimize the loss function. This can be interpreted as the least squares method for linear regression which minimizes the squared residuals. The loss function used for the neural network models in this paper is the Mean Squared Error (MSE) and is defined as:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

Where $\hat{Y}_i$ is the i-th ground truth value, Y_i is the i-th predicted value and n is the total number of samples.

Two positive properties of the MSE is that in general it is relatively computationally inexpensive and is sensitive to outliers because the difference between Y_i and $\hat{Y}_i$ is squared. This sensitivity to outliers has a positive influence on the used dataset because the outliers in the gas consumption represent valid data points, e.g. they are not corrupt data due to malfunctioning of the measurement devices.

Two other available loss functions are the Mean Absolute Percentage (MAPE) and the Symmetric Mean Absolute Percentage Error (SMAPE). MSE is chosen over MAPE and SMAPE because they are less sensitive for outliers. Where MAPE is defined as

$$\mathrm{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|\hat{Y}_i - Y_i|}{|Y_i|}$$

and SMAPE as

$$\text{SMAPE} = \frac{100\%}{2n}\sum_{i=1}^{n}\frac{|Y_i - \widehat{Y_i}|}{|\widehat{Y_i}| + |\widehat{Y_i}|}.$$

Notice how the MSE is scale dependent whereas MAPE and SMAPE are in percentages. The MAPE and SMAPE are still used as evaluation metrics, together with MSE to determine the performance of each model.

2.4 Optimizers and Learning rate scheduler

Like stated in chapter 2.3, neural networks use an optimizer to minimize the loss function. In this research Adam [16] and Nadam [17] are used for the models. The used optimizer is specified per model in their respective subchapters. The main difference between Adam and Nadam is that Adam is essentially RMSprop with momentum whereas Nadam is Adam RMSprop with Nesterov momentum. This is simplified by imagining a curved plane in $\mathbb{R}^3$. When trying to get to the lowest point of this plane, Nadam will jump over hills more quickly than Adam would by default [18]. Adam and Nadam are chosen instead of the commonly used Stochastic Gradient Descent (SGD) method, because they converge quicker than SGD does [19].

To improve the rate of convergence of the loss function, a cosine annealing learning rate scheduler with periodic restarts is applied to the optimizer. Together with improving the rate of convergence, this also gives the ability to find a lower and wider minimum [20].

Regularly, without the learning rate scheduler, the learning rate η_t is set to a fixed value for each batch within the total amount of epochs T_i. Whereas with the learning rate scheduler, the learning rate is changed per batch within the i-th run as follows [21]:

$$\eta_t = \eta_{min}^{i} + \frac{1}{2}(\eta_{max}^{i} - \eta_{min}^{i})(1 + \cos\left(\frac{T_{cur}}{T_i}\pi\right)$$

Where η_{max}^{i} and η_{min}^{i} are the ranges for the learning rate and T_{cur} is the number of epochs since the last restart.

2.5 Feature and batch normalization

The features X are standardized by removing the mean and scaling to unit variance using the 'StandardScaler' function from scikit-learn. This function scales all feature sampled to a distribution which is centered around 0. This prevents one or more features dominating the others. The models will also converge less quick and have a likelihood to have a lower accuracy when the features are not scaled. Each feature from the features train dataset X_{train} is scaled independently. The standard score Z of the feature sample is calculated as [22]:

$$Z = \frac{X_{train} - u}{s}.$$

Where u is the mean value of the sample and s is the standard deviation from the sample. The standard score is stored and used to also transform the features X_{test} from the test dataset. The standard score is not being recalculated for the test set. This prevents having data leakage from the distribution of the feature samples from the test set, to the train set.

2.6 One hot encoding

The hour of the day, day of the week and current season values are extracted as features from the timestamp of each row in the dataset. Where hour of day ranges from 0 to 23, day of the week from 0 to 6 and season from 1 till 4. These features are one hot encoded. This transforms the data so each value of the feature has a separate column in the dataset. Take for example this matrix.

$$\begin{bmatrix}0\\1\\2\end{bmatrix} \rightarrow \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$$

Where the day of the week is represented as a number in the column vector. After transforming this column vector to the 3×3 matrix, each column represents a day of the week. With 1 representing that at given row it is currently that day of the week. Where column one represents that row being on a Monday, column two being on a Tuesday and column three being on a Wednesday.

Representing the features extracted from the timestamp this way allows the models to assign different weights to for example 07:00 AM on a Monday and 09:00 AM on a Saturday.

2.7 Architecture evaluations

Hyperas is used to evaluate different architectures of each neural network model. A commonly used distribution of nodes and layers is set as the available parameter space. Hyperas is set to evaluate a fixed amount of possibilities from the parameter space. The amount of evaluations is chosen so the total evaluation time per model takes 24 hours. During each evaluation Hyperas trains a different architecture for a specific number of epochs. In the end the best performing architecture is used for the models in this paper. The amount of evaluations per model is defined in *Table 2*.

2.8 Initial neural network architecture setup

All neural network models have been programmed in Python using the Keras library with a TensorFlow backend and are trained on a NVIDIA GeForce 960m GPU. The weights of each layer are initialized by a truncated normal distribution. The bias from each dense layer is turned off because each layer is followed up by a batch normalization layer, apart from the output layer. This batch normalization layer normalizes the weights like described in chapter 2.5 and is also applied after recurrent and convolutional layers. Finally, each layer apart from the output layer is followed up by the Leaky version of a Rectified Linear Unit (LeakyReLu) activation function, which is defined as [23]:

$$h^{(i)} = \max\left(\boldsymbol{w}^{(i)T}, 0\right) = \begin{cases} \boldsymbol{w}^{(i)T}x & \boldsymbol{w}^{(i)T}x > 0 \\ 0.01\boldsymbol{w}^{(i)T}x & \text{else} \end{cases}$$

Where $\boldsymbol{w}^{(i)}$ is the weight vector for the i-th hidden node and x is the node input.

2.9 MVLR

MVLR is the simplest model used in this research. Despite it being a simple model, it is often being used for the energy forecasting and is known to make relatively accurate predictions [24, 8]. The combination of the performance and simplicity makes the evaluation metrics from MVLR the baseline that will be used to compare other models results with.

$$y = b_0 + b_1X_0 + b_2X_1 + b_3X_2 + \sum_{i=0}^{23} b_{4+i}X_{3+i} + \sum_{j=0}^{6} b_{28+j}X_{27+j} + \sum_{k=0}^{3} b_{34+k}X_{33+k}$$

where			
	y	gas consumption	$[\mathrm{m}^3/\mathrm{h}]$
	b_0	offset	[-]
	X_0	temperature	[°C]
	X_1	Wind speed	[m/s]
	X_2	rain intensity	[mm/h]
	X_{3+i}	hour of the day	[-]
	X_{27+j}	day of the week	[-]
	X_{33+k}	season	[-]

2.10 DNN

The next model used is a feed-forward DNN, this is one of the basic types of neural networks due to all connections going in one direction without cycles or loops. The data from the input layers are passed on to the next layers of nodes (hidden layers) and based on the weights, offset and activation function they compute a value per node, which is passed on to the next layer until the output node is reached. The value of the output node will then output a prediction y_{pred}. One of the main benefits of feed-forward DNN is the ability to adapt to non-linear relationships, in contrast to MVLR.

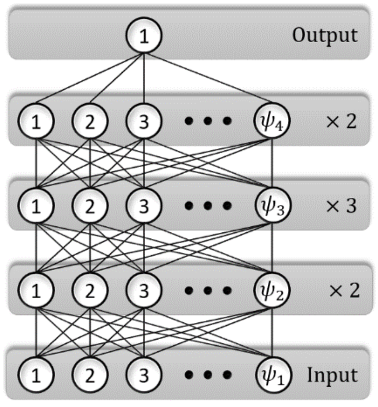
Output
1
1 2 3 ••• ψ_4 $\times 2$
1 2 3 ••• ψ_3 $\times 3$
1 2 3 ••• ψ_2 $\times 2$
1 2 3 ••• ψ_1 Input

Figure 3. Where $\psi_1, \psi_2, \psi_3, \psi_4$ represent the number of nodes of respective layer and are equal to 64, 256, 64, 1024, 8 respectively. The $\times 2$ represents this layer configuration being repeated two times behind each other.

Figure 3 contains a schematic of the used DNN architecture. The input shape of this model is matrix of shape (features), containing the features of the current hour, to predict the next target value of shape (target). Where (features) when using five features would be (5), e.g. a five-dimensional row vector. Simplified, the weather information from the current hour is used to predict the gas consumption of the next hour. Nadam is used as the optimizer while training this model.

2.11 LSTM and GRU

LSTM and GRU are based on recurrent neural networks (RNN), which are often used for natural language and text processing [25]. LSTM networks are different from RNNs by the ability to store historical information which has been processed in its internal memory units, which can be a big advantage when using time series data. [6]. Compared to LSTM, GRU networks use less parameters per node and can thus be interpreted as a simplified version of the LSTM model [26].

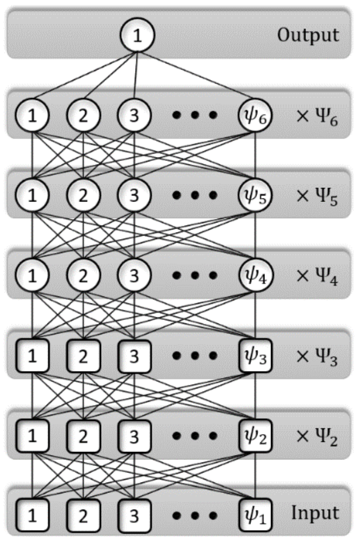
Output
1
1 2 3 ••• ψ_6 $\times \Psi_6$
1 2 3 ••• ψ_5 $\times \Psi_5$
1 2 3 ••• ψ_4 $\times \Psi_4$
1 2 3 ••• ψ_3 $\times \Psi_3$
1 2 3 ••• ψ_2 $\times \Psi_2$
1 2 3 ••• ψ_1 Input

Figure 4. Where $\psi_1, \ldots, \psi_6$ represent the number of nodes of respective layer. For LSTM these are equal to 8, 0, 16, 128, 8, 16 and for GRU are equal to 16, 8, 4, 0, 8, 8 respectively.
Each layer configuration being repeated Ψ_i times behind each other is represented by $\times \Psi_i$, where i is the layer number. For LSTM $\Psi_2, \ldots, \Psi_6$ are equal to 0, 1, 3, 2, 1 and for GRU are equal to 1, 1, 0, 4, 1 respectively.

The input shape of this model is a matrix of shape (timesteps, features), containing the features of all the timesteps, to predict the next target value of shape (target). Timesteps can be interpreted as the model being able to look back a specific number of hours. For both the LSTM and GRU models this is set 120 hours. Simplified and summarising, the weather information from the past 120 hours is used to predict the gas consumption of the next hour. The architectures used can be seen in *Figure 4*. Adam has been used as the optimizer during training of the LSTM model, whereas Nadam has been used for GRU.

2.12 CNN

A CNN is a type of deep neural network, most commonly used for image recognition. Partly due to the development of autonomous cars, image classification, facial recognition and more, CNNs are one of the most advanced neural networks currently being developed in computer science [27, 28, 29]. Compared to the previously discussed networks, a requirement to apply CNNs is the addition of a channel dimension to the feature matrix used in the RNN model. This changes the matrix from shape [timesteps, features] to [height, width, channel]. Where channel is the colour dimension, three for RGB images and one for grey-scaled images. Timesteps and features are interpreted as the height and width of the input image. The architecture used can be seen in *Figure 5*. The model has been trained with the Nadam optimizer.

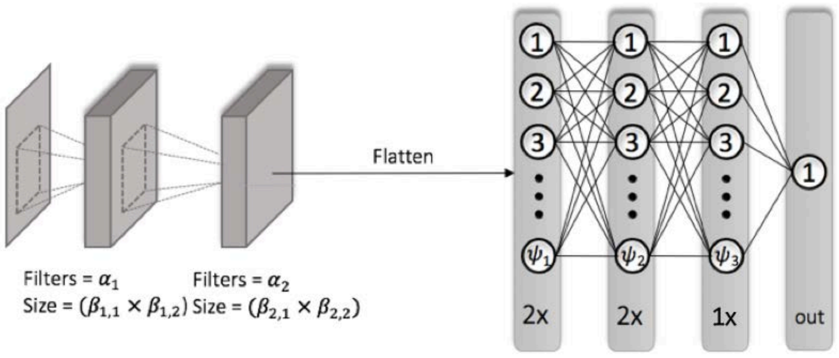
Flatten
Filters = α_1
Size = $(\beta_{1,1} \times \beta_{1,2})$
Filters = α_2
Size = $(\beta_{2,1} \times \beta_{2,2})$
2x
2x
1x
out

Figure 5. Where α_1 is equal to 5, α_2 is equal to 8, $(\beta_{1,1} \times \beta_{1,2})$ equals (8 x 4) and $(\beta_{2,1} \times \beta_{2,2})$ equals (10 x 8). The final output of the CNN is flatted and fed into a DNN where $\psi_1 \dots \psi_3$ equals 64, 128, 256 and $\Psi_1 \dots \Psi_3$ are equal to 2, 2, 1 respectively.

2.13 Time distributed CNN + RNN + DNN

The Time Distributed model consists of a time distributed CNN layer, being followed up by an LSTM and finally a DNN. This model combines the power of all three models. An image with (timesteps, columns) of (120, 39) is being fed into the input layer. This image is then reshaped to 24 smaller images of (5, 39, 1). Where 1 is the channel dimension required by the CNN.

Each smaller image is fed to the CNN and the flattened CNN output is saved in memory. These 5 flattened outputs make up the sequence that is being fed into the RNN of shape (5, flattened CNN output). From here on the RNN and DNN are applied as explained in their respective subchapters. Nadam is used as the optimizer to train this model.

3. RESULTS

As seen in *Table 2*, DNN performs best on an hourly resolution when looking at the MAPE. On a daily and weekly resolution, MVLR outperforms the other models. In all resolutions, LSTM has the lowest performance.

Table 2. The values of cross-validation evaluation metrics, amount of architecture evaluations and the amount of epochs done per model.

Model [-]	**Resolution**	**MSE [-]**	**MAPE [%]**	**SMAPE [%]**	**Architecture evaluations [-]**	**time per epoch [s]**	**Epochs [-]**
MVLR	Hour	0.62	78.3	193	n.a.	n.a.	n.a.
	Day	99.0	20.2	920			
	Week	$2.44 \cdot 10^3$	17.0	7.80			
DNN	Hour	0.67	50.1	16.6	$1.00 \cdot 10^3$	$4.00 \cdot 10^{-6}$	$3.50 \cdot 10^4$
	Day	104	25.1	10.5			
	Week	$2.96 \cdot 10^3$	20.1	8.70			
LSTM	Hour	1.00	139	33.9	50	$4.62 \cdot 10^{-3}$	$4.00 \cdot 10^3$
	Day	206	99.7	30.1			
	Week	$7.06 \cdot 10^3$	95.0	31.1			
GRU	Hour	1.19	78.6	30.5	100	0.11	$4.00 \cdot 10^3$
	Day	264	59.8	19.4			
	Week	$9.38 \cdot 10^3$	45.3	16.9			
CNN	Hour	0.84	84.3	28.3	50	0.76	$8.00 \cdot 10^3$
	Day	115	33.3	13.5			
	Week	$3.51 \cdot 10^3$	32.3	13.6			
Time Dist.	Hour	0.91	74.0	26.8	100	$2.88 \cdot 10^{-3}$	$4.00 \cdot 10^3$
	Day	184	42.7	16.4			
	Week	$5.93 \cdot 10^3$	41.5	16.3			

Figure 6 shows that the models tend to forecast systematically below the real values. Furthermore, during summer MVLR and DNN outperform LSTM, GRU, CNN and Time Distributed. During winter, the difference between the real and the forecasted values become larger. The hourly forecasts per model are found in *Appendix I*, *Figure 8-13*.

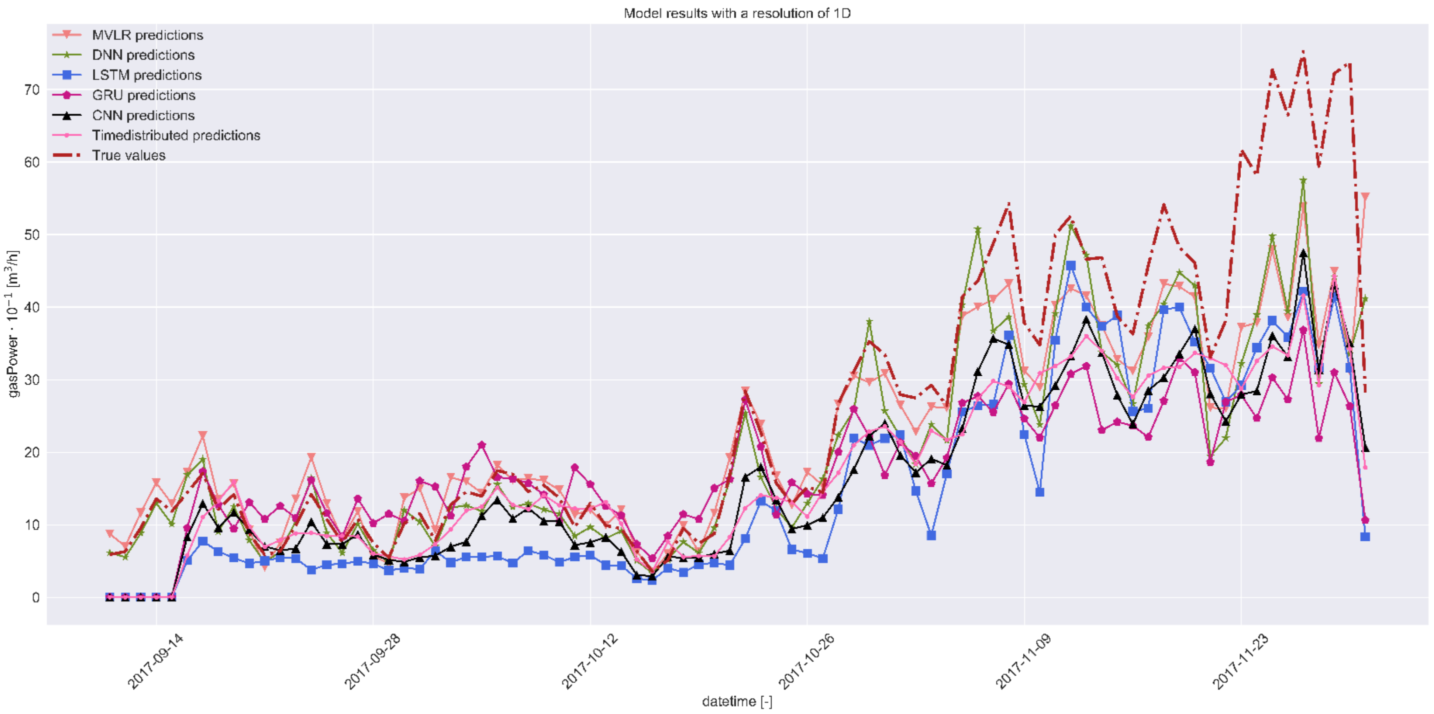

Figure 6. The forecasted gasPower consumption of the different models on a daily resolution.

4. CONCLUSION

This paper compares several machine learning algorithms to forecast the mean gas consumption of 52 dwellings, representing a block of dwellings on the aggregated level. Forecasts are done with an hourly resolution by using the windspeed, rain intensity, temperature, season, hour of day and day of the week as features. The feature selection is made on using as few features as possible, with the objective of keeping the computation power required as low as possible. The choice of models has been based on previous research studies which are mainly focused on forecasting the gas and electricity consumption in commercial and office buildings. Furthermore, three types of deep learning models have been combined into a single model called Time Distributed. Time Distributed combines the potential of CNNs and RNNs, with the goal of getting a better performance regarding the outcome of the results. More specifically, this is a time distributed CNN followed up by a LSTM and DNN.

To validate the applicability of each model, the models have been compared on performance and computational time required. The gas consumption data of the mean of 52 dwellings has been split into a training and test dataset of 70 % and 30 %, respectively. Predictions have been cross validated on the test set with an hourly resolution. To evaluate the performance on multiple resolutions, the hourly predictions have been down sampled to one day and one week resolution by summation.

Table 2 shows that on one hour resolution, the DNN model performs best with a 50.1 % MAPE, while LSTM has the lowest performance with a MAPE of 139 %. In comparison to MVLR, DNN outperforms MVLR because of its ability to adapt to non-linearities. DNN outperforms the other deep neural network models when comparing it to the other deep neural networks. Due to the 24 hour limitation on the architecture evaluations and number of epochs, DNN outperforms the other models thanks to its simplicity. A probable reason for LSTM and GRU performing worse than expected is the presence of NaNs in dataset. When NaNs are removed, missing timestamps affect to the periodicity of data and therefore could have an influence on the accuracy of the model. Furthermore, this could explain why one hot encoded features such as hour of the day, day of the week and season leads to performance gains. This is explained by the LSTM and GRU models adapting to a pattern of a fixed periodicity between the timesteps.

When comparing one day and one week resolutions, the results indicate that MVLR model outperforms the other models. MVLR has a MAPE of 20.2 % and 17.0 % , whereas LSTM is the lowest performing model with a MAPE of 99.7 % and 95.0 % on a one day and one week resolution respectively. When down sampling the data from one hour to lower resolutions, the cumulative error gets reduced because of the surface area of the errors getting smaller.

Further studies should focus on exploring the possibilities of getting more accurate results and applying the models on individual dwellings. One way the evaluation metrics could be improved is by using more data, e.g. a sampling period of full-year or more. This is substantiated on the variance between the validation and train loss. Improving the amount of training data could help with the recognition of human patterns and dependency on outside weather conditions. Alongside this, more features like the electricity consumption can be used to improve the accuracy of the deep learning models. However, this results in an increase in computational power needed, which can be a drawback in certain situations, e.g. when the hardware used has insufficient computational power. In the case of this research, as less features as possible have been used. In addition to the previously stated recommendations, increasing the amount of architecture evaluations, along with the number of epochs of the final model architecture, can lead to a better performance.

5. References

[1] Rijksoverheid, “Energieagenda: naar een CO_2-arme energievoorziening,” 07 12 2016. [Online]. Available: https://www.rijksoverheid.nl/documenten/rapporten/2016/12/07/ea.

[2] H. C. J. W. Y. G. Y. Y. Chengliang Xu, “Improving prediction performance for indoor temperature in public buildings based on a novel deep learning method,” 31 10 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360132318306875#bib6.

[3] F. A. S. F. Alberto Hernandez Neto, “Comparison between detailed model simulation and artificial neural network for forecasting building energy consumption,” 17 06 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778808001448.

[4] A. E. F. A. A. D.-s. K. M. A. R. C. J. K. Junghoon Woo, “A review of the state-of-the-art machine learning algorithms for building energy consumption prediction,” 05 2018. [Online]. Available: https://www.researchgate.net/publication/325742845_A_review_of_the_state-of-the-art_machine_learning_algorithms_for_building_energy_consumption_prediction.

[5] A. F. A. O. M. A. S. Salah Bouktif, “Optimal Deep Learning LSTM Model for Electric Load Forecasting using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches,” 21 05 21. [Online]. Available: https://www.researchgate.net/publication/325942309_Optimal_Deep_Learning_LSTM_Model_for_Electric_Load_Forecasting_using_Feature_Selection_and_Genetic_Algorithm_Comparison_with_Machine_Learning_Approaches.

[6] W.Kong, “Short Term residential load forecasting based on LSTM,” [Online]. Available: https://www.researchgate.net/publication/319886687_Short-Term_Residential_Load_Forecasting_based_on_LSTM_Recurrent_Neural_Network.

[7] Y. Y. G.-W. W. Ayşe Özmena, “Natural gas consumption forecast with MARS and CMARS models for residential users,” 19 01 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140988318300306.

[8] C. Jurado López, “Data-driven Predictive Control for Heating Demand in Buildings,” 24 03 2017. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3A15ee9126-0c03-46fd-988f-b7d583cc2398.

[9] W. Kong, “Short-Term Residential Load Forecasting based on LSTM Recurrent Neural Network,” 12 09 2017. [Online]. Available: https://www.researchgate.net/publication/319886687_Short-Term_Residential_Load_Forecasting_based_on_LSTM_Recurrent_Neural_Network.

[10] E. M. M. G. Paige A. Mynhoff, “Statistical Learning versus Deep Learning: Performance Comparison for Building Energy Prediction Methods,” 09 2018. [Online]. Available: https://www.researchgate.net/publication/327605408_Statistical_Learning_versus_Deep_Learning_Performance_Comparison_for_Building_Energy_Prediction_Methods.

[11] R. S. I. V. M. Z.-S. Kristian Sabo, “Mathematical models of natural gas consumption,” 03 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890410004863.

[12] KNMI, “Daggegevens KNMI Rdam,” 2018. [Online]. Available: http://projects.knmi.nl/klimatologie/daggegevens/selectie.cgi.

[13] O. K. E. P. M. M. Marek Brabec, “A nonlinear mixed effects model for the prediction of natural gas consumption by individual customers,” 22 10 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207008000976.

[14] J. Szoplik, “Forecasting of natural gas consumption with artificial neural networks,” 14 03 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S036054421500393X.

[15] D. M. Allen, “Mean Square Error of Prediction as a Criterion for Selecting Variables,” 2012. [Online]. Available: https://amstat.tandfonline.com/doi/abs/10.1080/00401706.1971.10488811#.XDYKmVxKhPZ.

[16] J. L. B. Diederik P. Kingma, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION,” 30 01 2017. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf.

[17] T. Dozat, “Incorporating Nesterov Momentum into Adam,” 2016. [Online]. Available: http://cs229.stanford.edu/proj2015/054_report.pdf.

[18] S. Ruder, “An overview of gradient descent optimization,” 15 06 2017. [Online]. Available: https://arxiv.org/pdf/1609.04747.pdf.

[19] S. Ruder, “An overview of gradient descent optimization algorithms,” June 2017. [Online]. Available: https://arxiv.org/abs/1609.04747.

[20] W. Lei, B. Shueh, “Stochastic Gradient Descent with Hyperbolic-Tangent Decay on Classification,” 2017. [Online]. Available: https://arxiv.org/pdf/1806.01593.pdf.

[21] F. H. Ilya Loshchilov, “SGDR: STOCHASTIC GRADIENT DESCENT WITH,” 03 05 2017. [Online]. Available: https://arxiv.org/pdf/1608.03983.pdf.

[22] T. Adeyemi, “The Effective use of Standard Scores for Research in Educational Management,” 25 08 2011. [Online]. Available: http://maxwellsci.com/print/rjms/v3-91-96.pdf.

[23] A. Y. H. A. Y. N. Andrew L. Maas, “Rectifier Nonlinearities Improve Neural Network Acoustic Models,” Computer Science Department, Stanford University, CA 94305 USA, [Online]. Available: https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

[24] O. M. S. N. Vincenzo Bianco, “Electricity consumption forecasting in Italy using linear regression models,” 04 06 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544209002539.

[25] M. Morchid, “Parsimonious memory unit for recurrent neural networks with application to natural language processing,” 24 05 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218306660.

[26] C. G. C. B. Junyoung Chung, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” 11 12 2014. [Online]. Available: https://arxiv.org/pdf/1412.3555.pdf.

[27] B. K.-F. F. T. Boukaye Boubacar Traore, “Deep convolution neural network for image recognition,” 12 10 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954118302140.

[28] M. B. L. S. M. W. Luca Caltagirone, “LIDAR–camera fusion for road detection using fully convolutional neural networks,” 04 11 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889018300496.

[29] H. R. Xiaolong Zhu. Meng Zhu, “Method of plant leaf recognition based on improved deep convolutional neural network,” 19 06 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389041717303480.

6. APPENDIX I

Table 3. The following table contains all the available weather and smart meter parameters, with a brief explanation on their meaning. The rows highlighted are the features used to feed the models and the darkest one is the target. *Have been manually calculated.

Parameter	Unit	Description	Sample rate
DD	deg	Wind direction	15 min
DR	s	Precipitation time	
FX	m/s	Maximum gust of wind at 10 m	
FF	m/s	Windspeed at 10 m	
N	okta	Cloud coverage	
P	hPa	Outside pressure	
Q	W/m^2	Global radiation	
RG	mm/h	Rain intensity	
SQ	min	Sunshine duration	
T	℃	Temperature at 1,5 m (1 minute mean)	
T10	℃	Minimum temperature at 10 cm	
TD	℃	Dew point temperature	
U	-	Relative humidity at 1,5 m	
VV	m	Horizontal sight	
WW	-	Weather- and station-code	
Timestamp	-	Timestamp of data telegram (set by smart meter) in local time	10 s
eMeter	kWh	Meter reading electricity delivered to client, normal tariff	
eMeterReturn	kWh	Meter reading electricity delivered by client, normal tariff.	
eMeterLow	kWh	Meter reading electricity delivered to client, low tariff	
eMeterLowReturn	kWh	Meter reading electricity delivered by client, low tariff	
ePower	kWh	Actual electricity power delivered to client	
ePowerReturn	kWh	Actual electricity power delivered by client	
gasTimestamp	-	Timestamp of the gasMeter reading (set by smart meter) in local time	1 h
gasMeter	m^3	Last hourly value (temperature converted), gas delivered to client	
gasPower	m^3/h	Difference between current and previous gasMeter value *	
hour of day	-	Hour of day from the timestamp *	
day of week	-	Day of week when sample has been acquired *	
season	-	Season of the year when sample has been acquired *	

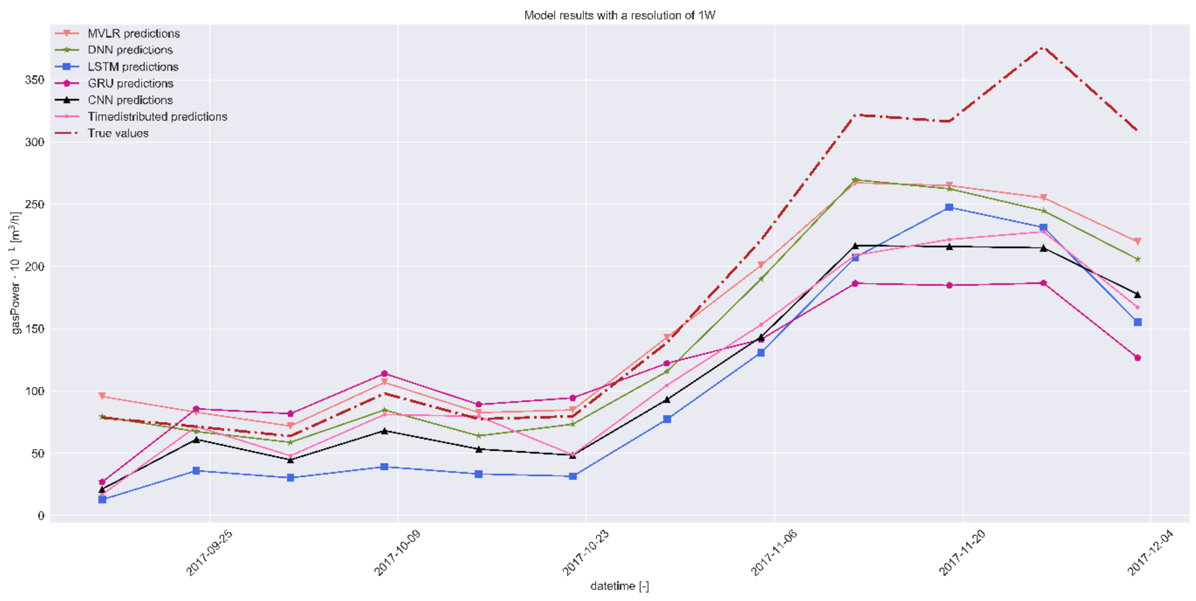

Figure 7. The forecasted gasPower consumption of the different models on a weekly resolution.

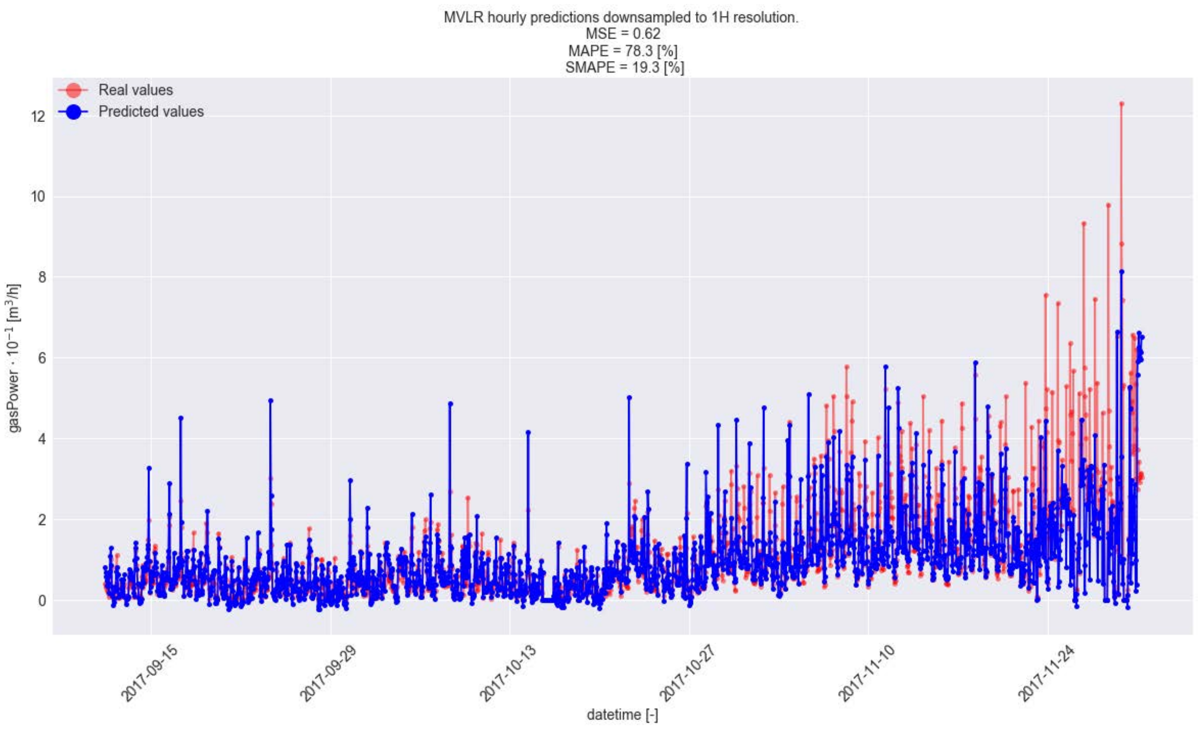

Figure 8. MVLR hourly results

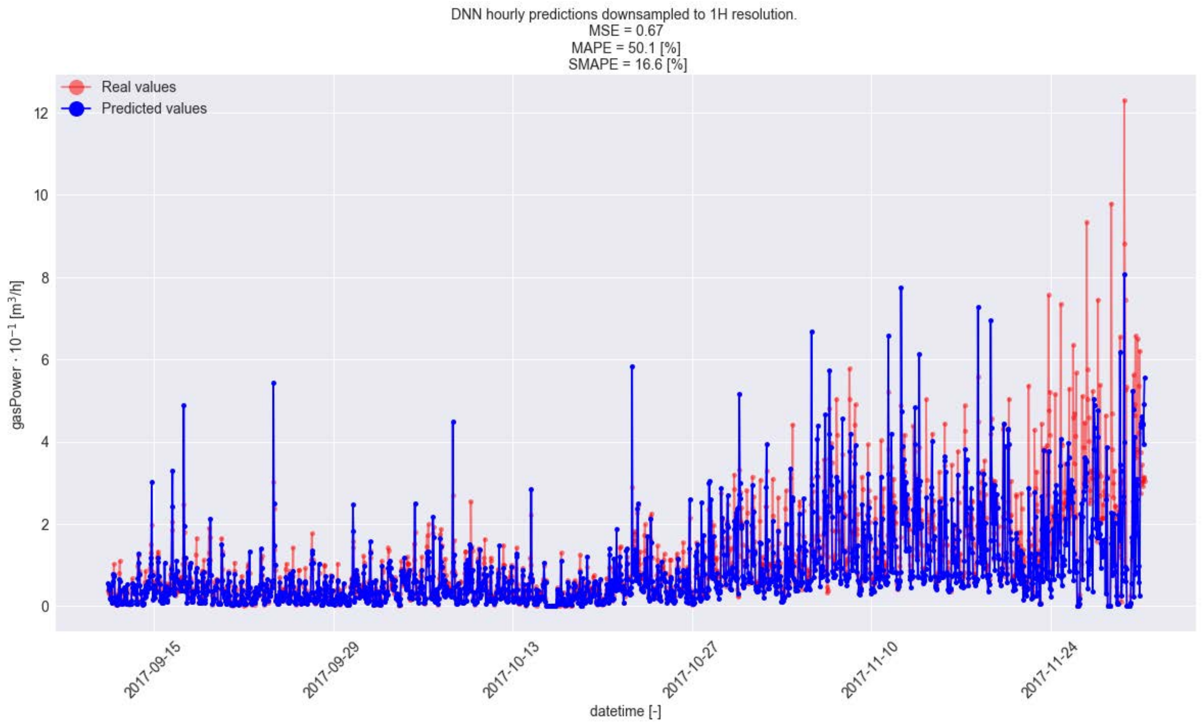

Figure 9. DNN hourly results

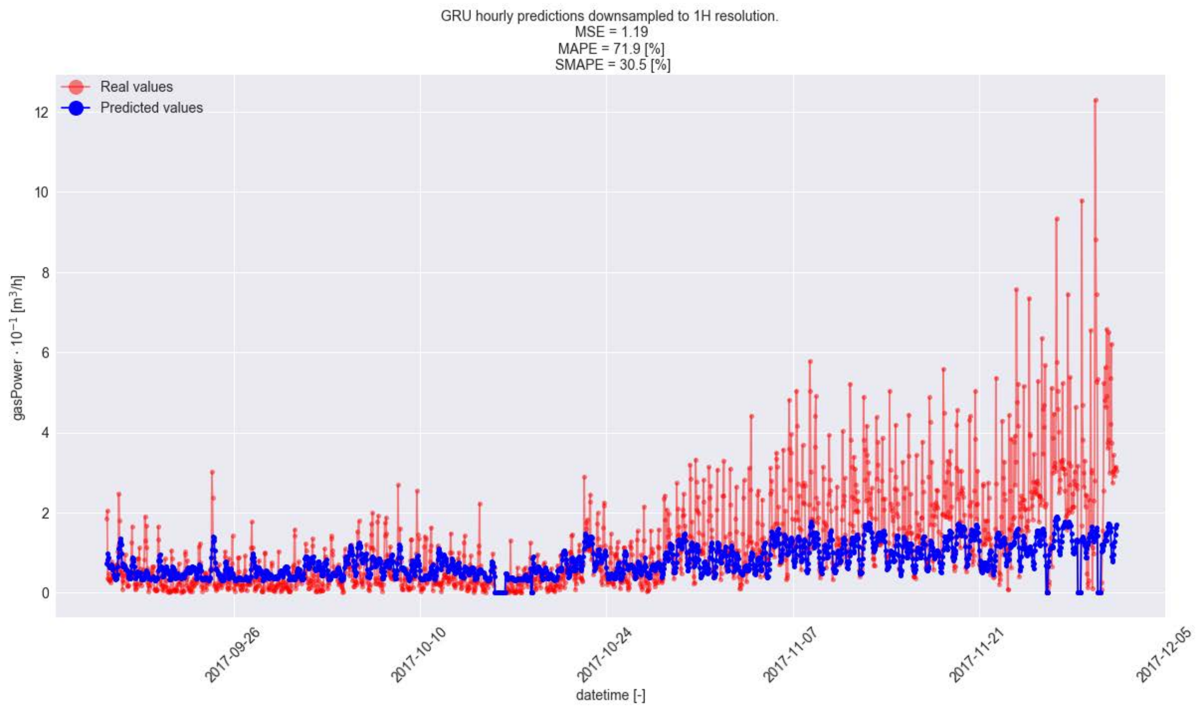

Figure 10. GRU hourly results

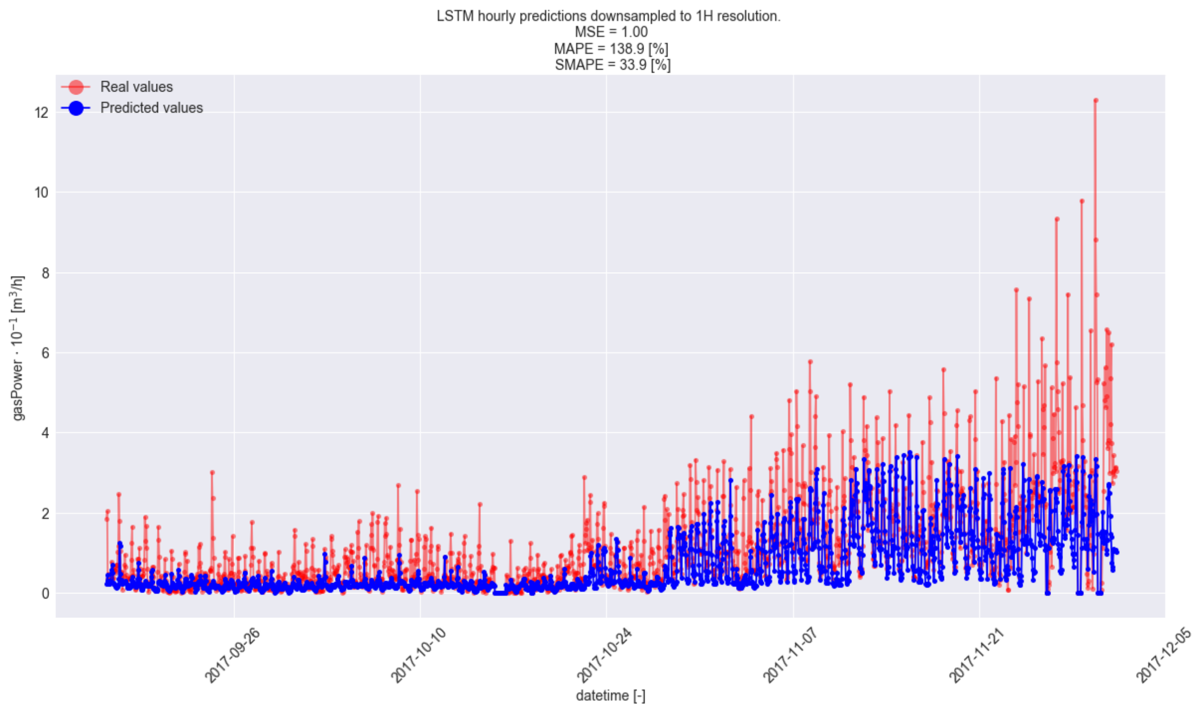

Figure 11. LSTM hourly results

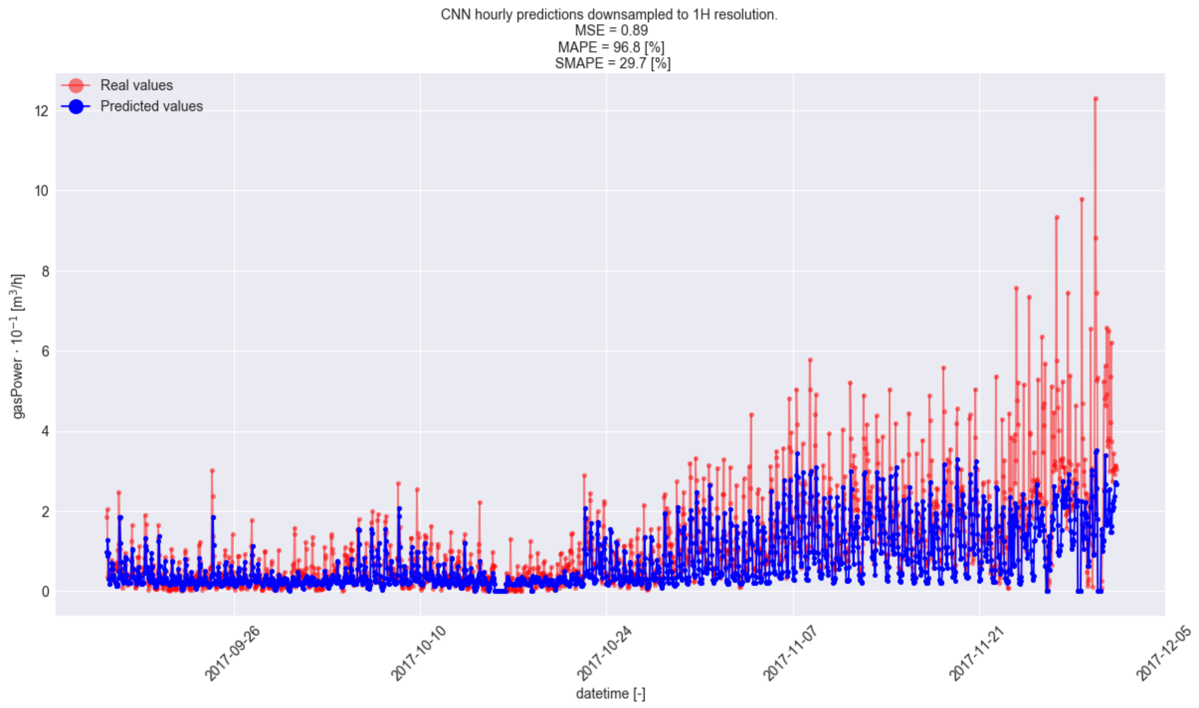

Figure 12. CNN hourly results

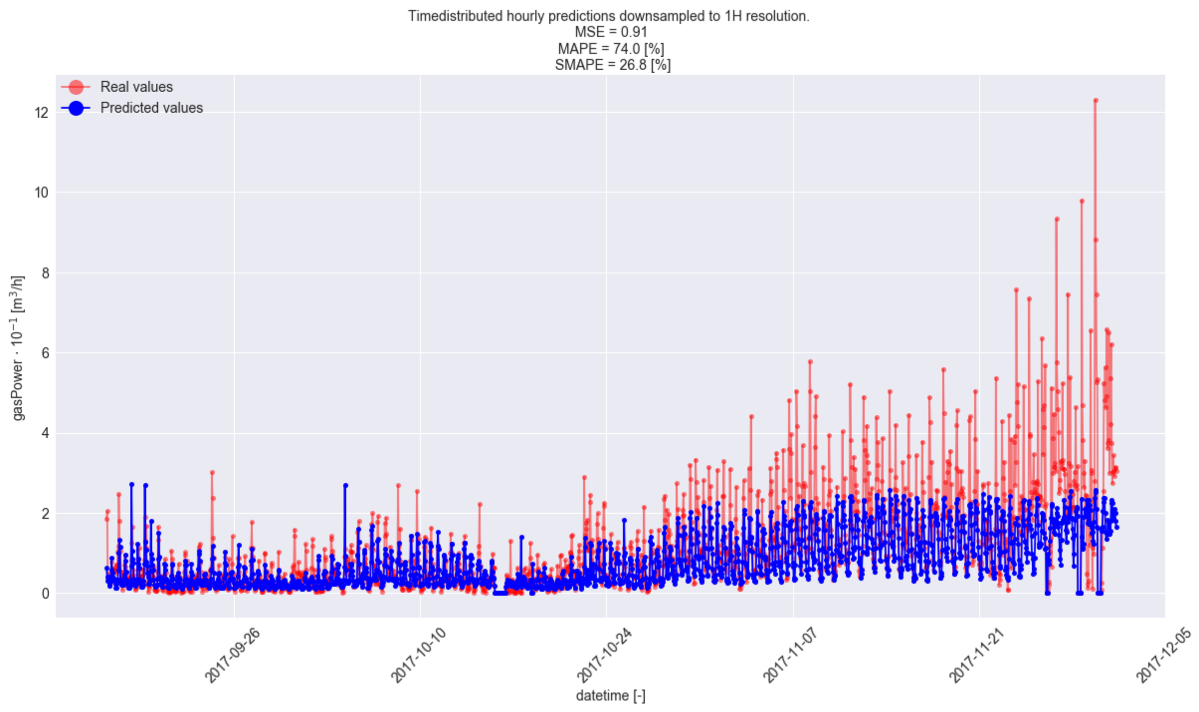

Figure 13. DNN hourly results

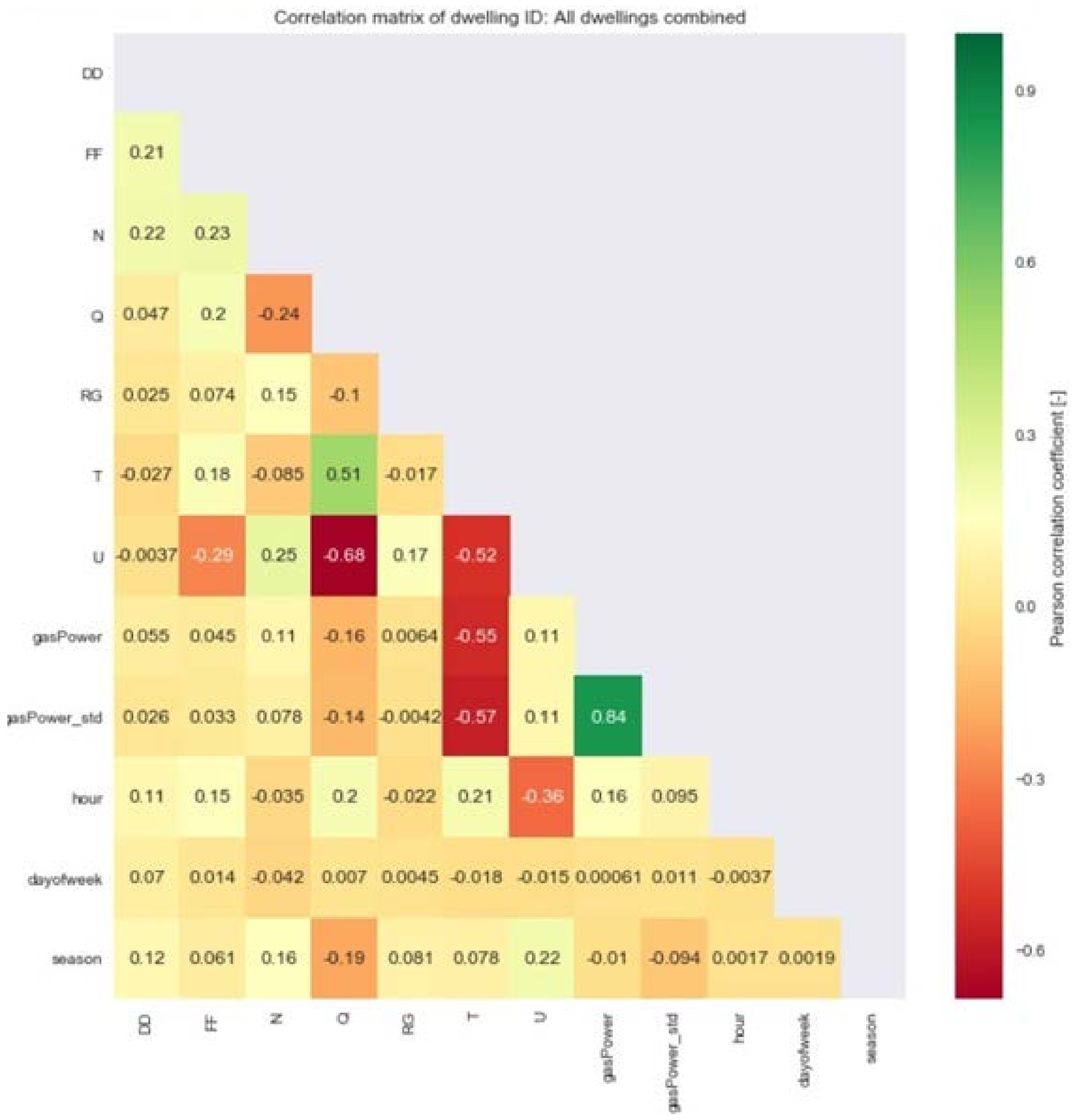

Figure 14. The Pearson correlations of all the available parameters.